PUBLIC KEY CRYPTOGRAPHIC METHODS AND SYSTEMS

Background of the Invention

(1) Field of the Invention

The present invention relates generally to cryptography and, more particularly, to public key cryptographic systems such as RSA.

(2) Description of the Prior Art

With the enormous volume of data that is transmitted electronically throughout the world, methods for securing the privacy of that data are crucial to the economy. Before the 1970s, senders and recipients would need to agree on some sort of secret key in order to encrypt messages such that they could not be deciphered by unauthorized third parties but could still be read by the intended recipient. This sort of symmetric cryptography alone is inconvenient in the Internet age, where it is not always easy to arrange a meeting to exchange a secret password that will allow for future secure communications. Fortunately, public key cryptography was developed in the last few decades by Diffie, Hellman, Rivest, Shamir, and Adelman, among others.

Public key cryptography allows for the secure exchange of information between senders and recipients without the necessity that the two parties first exchange a secret key. The recipient simply makes his public key available, which can be used by anyone to encrypt a message to him. Once a message is encrypted using the recipient's public key, only the private key can be used to restore the message to its original state. Only the recipient knows his private key, so messages encrypted with the public key are secure.

The standard methods for public key cryptography were developed by Rivest, Shamir, and Adelman (RSA), described in US Patent number 4405829. RSA and its

variants provide for encryption of data using a public key and decryption using a private key.

RSA security has been publicly and commercially used for communicating or transmitting information, data, documents, messages, and files; however, it is relatively slow (especially the process of decryption) and computationally intensive. This presents problems in many implementations, including servers that receive a large number of requests and mobile devices that have a small amount of computing resources available to them. The slow speed of RSA is a result of the large numbers required to ensure the security of the algorithm. The RSA scheme capitalizes on the extreme difficulty of factoring a large composite number into its constituent primes.

RSA and CRT RSA

RSA consists of three steps: key generation, encryption, and decryption.

Key Generation

Key generation starts by deciding on an adequate length for what is called the public modulus N. This choice is dictated by the difficulty of factoring N into its prime factors. Right now, N of length 1024 bits is considered a sufficient size to prevent factoring. The bit length of N will continue to go up in the future. Next, two random prime numbers that are each half the length of N, p and q, are generated. Next, a small odd integer, e, is selected such that e is relatively prime to lcm(p-1, q-1). In practice, e is usually chosen to be 65537. In this paper, we will refer to e as the public exponent and N as the public modulus. The RSA public key consists of the two integers (e, N).

The private exponent, d, is a multiplicative inverse of e(mod lcm(p-1, q-1)), so that $e*d = 1 \bmod (\mathrm{lcm}(p-1, q-1))$. Often, the private key refers to the set of numbers (p,q,d), so d should be referred to as the private exponent rather than as the private key.

Encryption

To encrypt message X using an RSA public key {e, N}, one must first convert X into an integer M using a formatting operation. Encryption of M into ciphertext C is then accomplished by calculating C as the remainder after N is divided into M taken to the power of e. In equation form, $C = M^e \bmod N$ where M is an integer greater than –1 and less than N, $0 \leq M < N$.

Decryption

To decrypt using the original implementation of RSA, M is obtained by calculating the remainder after N is divided into C taken to the power of d. In equation form, $M = C^d \bmod N$. M is then converted back to X by reversing the same formatting operation that was used to obtain M from X originally.

It is standard practice now to use the Chinese Remainder Theorem (CRT) for RSA decryption. Rather than compute $M = C^d \bmod N$, one calculates $d_p = d \bmod (p - 1)$ and $d_q = d \bmod (q - 1)$. Then, one calculates $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$. Then, one uses CRT to calculate M from M_p and M_q. This is about four times as fast as calculating $M = C^d \bmod N$ directly. For the remainder of this paper, we will refer to this method of RSA decryption as CRT RSA.

Since CRT RSA, a handful of improvements to the RSA methodology have been made to increase decryption speed. We will touch on each of these methods briefly, with

more attention paid to Multi-Prime and Multi-Power RSA, which are more in the field of the present invention.

Multi-Prime RSA

This method is detailed in US Patent number 5848159. Multi-Prime RSA suggests the use of more than two distinct prime factors to generate the public modulus N, whereas the RSA method traditionally uses only two distinct prime factors. For a modulus N of length 1024 bits, Multi-Prime RSA chooses three prime numbers p, q, r that are each one third the length of N. The encryption process i s exactly the same as traditional RSA. The decryption process for Multi-Prime RSA is relevantly similar to that of CRT RSA, except that three or more distinct prime numbers are used instead of two. In Multi-Prime RSA, like in traditional and CRT RSA, all of the distinct prime factors of the modulus N are used for decryption of messages.

Using multiple prime factors for RSA decryption increases the total number of calculations that need to be performed, but each calculation is less intensive since each prime factor is smaller than in the two-prime implementation. The result is a theoretical speedup of $b^2/4$, where b is the number of prime factors used. With N of length 1024 bits and b set to 3 (the current maximum for security reasons), Multi-Prime RSA achieves a theoretical speedup of 2.25 over two-factor CRT RSA methods.

Multi-Power RSA

This method is detailed in United States Patent Application 20020064278. This method is similar to the Multi-Prime method, except that the Multi-Prime method assumes that all of the prime numbers that make up the composite number N are distinct numbers. The Multi-Power method assumes that N is made up of more than two prime factors, but that

N is only made up of exactly two distinct prime factors. So, if $N = p*q*r$, it is assumed that $p = q$ and therefore $N = p^2r$. Like Multi-Prime RSA, encryption is performed in exactly the same was as traditional RSA. For decryption, Multi-Power RSA is able to capitalize on the fact that there are only two distinct prime numbers. So, only two large modular exponentiation computations are necessary (there are several smaller mathematical operations involved using this technique which are computationally negligible), providing increased efficiency over the Multi-Prime method. In Multi-Power RSA, like Multi-Prime and CRT RSA, all of the distinct prime factors of the modulus N are used for decryption of messages.

The result is a theoretical speedup of $b^3/8$, where b is the number of prime factors used. With N of length 1024 bits and b set to 3 (the current maximum for security reasons), Multi-Power RSA achieves a theoretical speedup of 3.375 over two-factor CRT RSA methods.

Batch RSA

Batch RSA is based on the idea that, in certain situations, two or more decryptions can be performed for the time cost of one. In order for this to work, very small public exponents must be used (such as 3 or 5). Further, the system only works if encryption is performed using different public exponents but the same public modulus. Further discussion of this technique is beyond the scope of this paper, but it is another example of a technique to speed up RSA decryption. With N of length 1024, it speeds up decryption by a factor of 2 or 3. However, there are many practical drawbacks to batch RSA techniques.

Rebalanced RSA

In standard RSA, encryption is much faster than decryption. In many applications, it would be desirable to change this behavior. Rebalanced RSA simply shifts more of the burden to the encryption process by increasing the size of the public exponent e (which is usually set to 65537), and tries to decrease the size of the private exponents as much as possible without creating security problems. Rebalanced RSA can be used in conjunction with many of the other methods here, such as Multi-Prime or Multi-Power RSA.

Hardware

There are also hardware approaches to speeding up RSA decryption. These methods consist of designing special processors or other hardware that is designed specifically for the type of modular arithmetic operations that RSA requires. Most hardware methods can be used in conjunction with the algorithmic methods described above.

Despite the existence of several software and hardware implementations for improving the overall speed of the decryption process in public key cryptography, there is still a need for continued improvements to the existing body of work. Faster decryption algorithms provide direct value in many situations, since companies that formerly needed two or three servers to handle decryption may now only need one etc. Given the fact that nearly all secure Internet traffic (SSL) current utilizes public key encryption/decryption, the present invention will provide great utility to the market.

Summary of the Invention

The present invention discloses methods for improving the computational efficiency and overall capabilities of RSA and related public key cryptographic systems. In prior art, decryption of messages that are encrypted using a public key {e, N} is always achieved using all of the distinct prime factors of N. For example, the private exponent d is usually calculated such that e*d = 1 mod Z, where Z is the product of each of the prime factors of N minus 1. In addition, RSA implementations that employ the Chinese Remainder Theorem (CRT) also use each of the distinct prime factors of N as moduli in small modular exponentiations that are combined to perform decryption. The present invention discloses several methods where the prime factors used for decryption are a proper subset of the prime factors in the modulus N, or the prime factors of the modulus used for encryption are a proper superset of the prime factors used for decryption. The techniques proposed in the present invention not only provide for a substantial increase in the computational speed of RSA decryption, but also open the door for exciting new applications of public key cryptography.

Accordingly, one aspect of the present invention is to provide a public key cryptosystem having a predetermined number of prime factors used for the generation of a modulus N and an exponent e, which may be generated in a variety of ways, wherein a proper subset of the prime factors of the modulus N, along with the exponent e, are required to decrypt messages encrypted using the public exponent e and the public modulus N, where e and N are calculated using RSA methods, and encryption occurs using RSA methods.

Another aspect of the present invention is to provide a method for encrypting/decrypting messages including the steps of: providing a public key cryptosystem having a predetermined number of prime factors used for the generation of a modulus N and an exponent e; wherein less than all of the prime factors of the modulus N are required to decrypt messages, documents, files, and/or data capable of being encrypted using the public exponent e and the public modulus N, where e and N are calculated using RSA methods, and encryption occurs using RSA methods.

Yet another aspect of the present invention is to provide a public key cryptosystem where a set of prime numbers S are sufficient for decryption of messages that are encrypted using a modulus N_p, calculated as the product of a set of numbers that is a proper superset of S, and a public exponent e, where encryption occurs with standard RSA methods using the e and N_p.

Still another aspect of the present invention provides a method for encrypting/decrypting messages where a single prime number, p, and a public exponent e are used to decrypt messages encrypted using a modulus N_p, calculated as the product of a set of numbers that is a proper superset of p, and the public exponent e, where encryption occurs with standard RSA methods using e and N_p.

Still another aspect of the present invention provides a method for encryption with a public key {e, N} where a plaintext message M is encrypted into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod (N*X)$, where N is the public modulus and X is any integer greater than 1.

Still another aspect of the present invention provides a method for decryption of a message that has been encrypted with the public key {e, N} where a ciphertext message

C is decrypted into a plaintext message M using any method that produces a value equivalent to $M = C^d \bmod N_d$, where N_d is the product of less than all of the prime factors of the public modulus N and d satisfies the equation $e*d = 1 \bmod Z$, where Z is the product of each of the k prime factors of N_d minus 1, $(p_1 - 1)*\ldots(p_k - 1)$.

These and other aspects of the present invention will become apparent to those skilled in the art after a reading of the following description of the preferred embodiment when considered with the claims.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE PRESENT INVENTION

For the present invention, the term messages is defined and used herein to include information, documents, files, and/or data that are preferably, but not necessarily, in electronic form.

The present invention provides a system for encrypting and/or decrypting messages, preferably electronic messages, including a public key cryptosystem where decryption is performed using less than all of the prime factors of the modulus used for encryption, along with the public exponent e.

One embodiment of the present invention provides a method for encrypting/decrypting messages including the steps of: providing a public key cryptosystem having a predetermined number of prime factors used for the generation of a modulus N and an exponent e; wherein less than all of the prime factors of the modulus N are required to decrypt messages capable of being encrypted using the public exponent e and the public modulus N, where e and N are calculated using RSA methods, and encryption occurs using RSA methods.

Another embodiment provides a method for encrypting/decrypting messages including the steps of:

Encrypting a plaintext message M into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod N$, where $0 \leq M < N_d$, such that the ciphertext C can be decrypted into the plaintext message M using only e and the prime factors of N_d

N being the product of all of the numbers in the set S;

S being a set of at least two prime numbers, $p_1 \ldots p_k$, where k is an integer greater than 1;

e preferably being a small prime number, alternatively e being a number that is relatively prime to the product of each distinct prime factor of N minus 1, $(N_1 - 1)^* \ldots (N_j - 1)$ for distinct prime factors of N 1 to j, where j is the number of distinct prime factors in N;

S_d being a proper subset of S;

N_d being the product of all of the numbers in the set S_d.

Preferably, a method for decrypting encrypted messages comprising the steps of:

determining if a derived modulus N_d is a squarefree number, meaning that its prime decomposition contains no repeated factors, and if so,

decrypting ciphertext C into message M using any method that produces a value equivalent to $M = C^d \bmod N_d$, where d is generated using the following steps:

Calculating the number Z_d as the product of each prime factor of N_d minus 1, $(N_{d1} - 1)^* \ldots (N_{dj} - 1)$ for prime factors of N_d 1 to j, where j is the number of prime factors in N_d;

generating the exponent d such that the following relationship is satisfied: $e*d = 1 \bmod Z_d$.

In one embodiment, the plaintext message M is restored by directly calculating $M = C^d \bmod N_d$.

In another embodiment, M is restored using the steps of:

calculating separate decryption exponents $d_{nd1} \ldots d_{ndj}$ for all prime factors of N_d 1 to j, where j is the number of prime factors in N_d so that the following relationship is satisfied for each member of N_d: $e*d_{ndi} = 1 \bmod (N_{di} - 1)$; and

performing decryptions of the form $M_i = C^{d_{ndi}} \bmod N_{di}$ for all prime factors of N_d from 1 to j, where j is the number of prime factors in N_d, and then using the values of each M_i and N_{di} to reconstruct M. These steps for reconstructing M from each value of M_i and N_{di} are preferably performed using the Chinese Remainder Theorem and/or Garner's algorithm, such as set forth in the references *C. Vuillame* <u>Efficiency Comparison of Several RSA Variants</u>, which are incorporated herein by reference in their entirety.

Alternatively, a method for decrypting encrypted messages including the steps of: decrypting the ciphertext message C to the plaintext message M by determining if the derived modulus N_d is squareful number, meaning that its prime decomposition contains at least one square, and if so;

calculating separate decryption exponents $d_{nd1} \ldots d_{ndj}$ for all distinct prime factors of N_d 1 to j, where j is the number of distinct prime factors in N_d so that the following relationship is satisfied for each distinct member of N_d: $e*d_{ndi} = 1 \bmod (N_{di} - 1)$;

for each distinct prime factor of N_d, N_{di}, calculating a value b_{di} as the number of times that N_{di} occurs as a prime factor in N_d;

calculating M_i for each distinct prime factor of N_d, N_{di};

and using all values of M_i, N_{di}, d_{ndi}, and b_{di} to restore the plaintext message M. The method of calculating M_i for each distinct prime factor of N_d, N_{di} is preferably performed using Hensel Lifting, such as set forth in the references *C. Vuillame* Efficiency Comparison of Several RSA Variants, which are incorporated herein by reference in their entirety. The restoration of the plaintext message M from the values of M_i, N_{di}, d_{ndi}, and b_{di} is preferably performed using techniques such as the Chinese Remainder Theorem and/or Garner's algorithm.

Another embodiment of the present invention includes a method for encrypting/decrypting messages, including the steps of:

Encrypting a plaintext message M into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod N_p$, where $0 \leq M < N$, such that the ciphertext C can be decrypted into the plaintext message M using e and the prime factors of N

N being the product of all of the numbers in the set S;

S being a set of at least one prime number, $p_1 \ldots p_k$, where k is an integer greater than 0;

S_p being a proper superset of S;

N_p being the product of all of the numbers in the set S_p;

e preferably being a small prime number, or alternatively, e being a number that is relatively prime to the product of each distinct prime factor of N_p minus 1, $(N_{p1} - 1)*\ldots(N_{pj} - 1)$ for distinct prime factors of N_p 1 to j, where j is the number of distinct prime factors in N_p;

Preferably, a method for decrypting encrypted messages including the steps of:

Decrypting the ciphertext message C into the plaintext message M by determining if the modulus N is a squarefree number; and if so then,

decrypting ciphertext C into message M using any method that produces a value equivalent to $M = C^d \bmod N$, where d is generated using the following steps:

Calculating the number Z as the product of each prime factor of N minus 1, $(N_1 - 1)*\ldots(N_j - 1)$ for prime factors of N 1 to j, where j is the number of prime factors in N;

then generating the decryption exponent d such that the following relationship is satisfied: $e*d = 1 \bmod Z$;

In one embodiment, M is directly calculated as $M = C^d \bmod N$.

In an alternative embodiment, M is calculated including the steps of:

calculating separate decryption exponents $d_1\ldots d_j$ for all prime factors of N 1 to j, where j is the number of prime factors in N so that the following relationship is satisfied for each member of N: $e*d_i = 1 \bmod (N_i - 1)$; and

performing decryptions of the form $M_i = C^{d_i} \bmod N_i$ for all prime factors of N from 1 to j, where j is the number of prime factors in N. Next, the values of each M_i and N_i are used to reconstruct M, preferably using the Chinese Remainder Theorem and/or Garner's algorithm.

Alternatively, a method for decrypting encrypted messages comprises the steps of:

Decrypting the ciphertext message C to the plaintext message M by determining if the derived modulus N is squareful number; if so then, calculating separate decryption exponents $d_{n1}\ldots d_{nj}$ for all distinct prime factors of N 1 to j, where j is the number of

distinct prime factors in N so that the following relationship is satisfied for each distinct member of N: $e*d_{ni} = 1 \bmod (N_i - 1)$;

for each distinct prime factor of N, N_i, calculating a value b_i as the number of times that N_i occurs as a prime factor in N;

Preferably using Hensel Lifting to calculate M_i for each distinct prime factors of N, N_i;

and preferably using the Chinese Remainder Theorem and/or Garner's algorithm to use each value of M_i, N_i, b_i and d_{ni} to restore the plaintext message M;

In still another embodiment of the present invention, encrypting/decrypting messages including the steps of:

Encrypting a plaintext message M into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod N_p$, where $0 \leq M < N$, such that the ciphertext C can be decrypted into the plaintext message M using e and the prime factors of N

N being the product of all of the members of set S;

S being a set of at least two numbers, $p_1 \ldots p_k$ where k is an integer greater than 1 and all members of S are equal to p_s, which is a prime number;

S_p being a superset of S;

N_p being the product of all of the numbers in the set S_p;

e preferably being a small prime number, or alternatively, e being a number that is relatively prime to the product of all of the distinct prime factors of N_p minus 1, $(N_{p1} - 1)*\ldots(N_{pj} - 1)$ for distinct prime factors of N_p 1 to j, where j is the number of distinct prime factors in N_p;

Preferably , decryption of encrypted messages includes the steps of:

Decrypting the ciphertext message C to the plaintext message M by:

Calculating b as the number of times that the number p_s occurs as a prime factor in N;

Generating an exponent d such that the following equation is satisfied:

$e*d = 1 \bmod (p_s - 1)$;

Using Hensel Lifting to transform C into M with d, p_s, and b as input values.

In still another embodiment of the present invention, a method for crytographic communications is disclosed, including the steps of:

Encrypting a plaintext message M into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod N_p$, where $0 \leq M < p$, such that the ciphertext C can be decrypted into the plaintext message M using e and p

p being a prime number;

S being a set containing only the number p;

S_p being a superset of S;

N_p being the product of all members of the set S_p;

e preferably being a small prime number, or alternatively, e being a number that is relatively prime to the product of each distinct prime factor of N_p minus 1, $(N_{p1} - 1)*\ldots(N_{pj} - 1)$ for distinct prime factors of N_p 1 to j, where j is the number of distinct prime factors in N_p;

Preferably, decryption of encrypted messages comprises the steps of:

Decrypting using any method that produces a value equivalent to as $M = C^d \bmod p$, where d is generated using the following step:

Calculating d such that the following equation is satisfied:

$e*d = 1 \bmod (p - 1)$;

In another embodiment of the present invention, a method for establishing cryptographic communications is disclosed, including the steps of: calculating a composite number N, which is formed from the product of distinct prime numbers S, $p_1, \ldots p_k$ where $k \geq 1$. Encoding a plaintext message M, to a ciphertext message C, where M corresponds to a number representative of a message and $0 \leq M < S$; generating an exponent e, which is preferably a small prime number but can alternatively be established as any number that is relatively prime to the product of each distinct prime factor of N minus 1, $(N_1 - 1), \ldots (N_j - 1)$ for distinct prime factors of N 1 to j, where j is the number of distinct prime factors in N; transforming said plaintext, M, into said ciphertext, C, where C is developed using any method that produces a value equivalent to $C = M^e \bmod N$, such that ciphertext C can be decrypted into plaintext M using only e and S.

Decryption of messages encrypted in such a way is performed by: decoding the ciphertext message C to the plaintext message M, wherein said decoding includes the step of: transforming said ciphertext message C to plaintext M, using any method that produces a value equivalent to $M = C^d \pmod S$, where d is generated such that $e*d = 1 \bmod (S - 1)$;

Another embodiment of the present invention discloses a system for encrypting and decrypting electronic communications including a network of computers and/or computer-type devices, such as personal data assistants (PDAs), mobile phones and other devices, in particular mobile devices capable of communicating on the network;

generating at least one private key and at least one public key, wherein the at least one private key is determined based upon any one of a multiplicity of prime numbers that when multiplied together produce N, which is the modulus for at least one of the public keys.

Another embodiment of the present invention discloses a method for public key decryption where less than all of the distinct prime factors of a number N are used to decrypt a ciphertext message C into plaintext message M, where encryption occurs with the public key {e, N} using any method that produces a value equivalent to $C = M^e \bmod N$.

Another embodiment of the present invention discloses a method for public key encryption with a public key {e, N} where a plaintext message M is encrypted into a ciphertext message C using any method that produces a value equivalent to $C = M^e \bmod (N*X)$, where N is the public modulus and X is any integer greater than 1.

Another embodiment of the present invention discloses a method for public key decryption of a message that has been encrypted with the public key {e, N} where a ciphertext message C is decrypted into a plaintext message M using any method that produces a value equivalent to $M = C^d \bmod N_d$, where N_d is the product of less than all of the prime factors of the public modulus N and d satisfies the equation $e*d = 1 \bmod Z$, where Z is the product of each of the k prime factors of N_d minus 1, $(p_1 - 1)*\ldots(p_k - 1)$.

Yet another embodiment of the present invention discloses a method for public key decryption of a message that has been encrypted using any method that produces a value equivalent to $C = M^e \bmod N$, where a ciphertext message C is decrypted into a plaintext message M using any method that produces a value equivalent to $M = C^d \bmod$

N_d, where N_d is the product of less than all of the prime factors of the public modulus N and d satisfies the equation $e*d = 1 \bmod Z$, where Z is the product of each of the k prime factors of N_d minus 1, $(p_1 - 1)*\ldots(p_k - 1)$.

Generally, decryption methods according to the present invention are paired with particular encryption steps such that decryption is accomplished using the encryption exponent e and less than all of the prime factors of the encryption modulus, as set forth in the following design examples, which are provided for the purpose of illustrating methods of the present invention, without limiting it expressly thereto.

<u>Design Examples</u>

<u>Example #1</u>

Generating prime numbers p and q as the members of set S, and calculating $N = p*q$. It is preferred that p is set to the minimum bit length, given existing security constraints and the expected message size, and that q is set to a bit length such that the bit length of N reaches its recommended size.

Calculating e as a small prime number, such as 65537.

Including p as the only member of the proper subset, S_d.

Setting $N_d = p$.

Calculating the private exponent d such that $e*d = 1 \bmod (p - 1)$.

Encrypting plaintext M into ciphertext C as $C = M^e \bmod N$, where $0 \leq M < N_d$.

Decrypting ciphertext C into plaintext M as $M = C^d \bmod N_d$.

<u>Example #2</u>

Generating prime number p as the only member of set S, and setting $N = p$.

It is preferred that p is set to the minimum bit length given existing security constraints and the expected message size.

Calculating e as a small prime number, such as 65537.

Creating the set S_p as a proper superset of set S containing members p and q, and calculating $N_p = pq$. It is preferred that q is large enough so that the bit length of the N_p reaches its recommended size.

Calculating the private exponent d such that $e*d = 1 \bmod (p - 1)$.

Encrypting plaintext M into ciphertext C as $C = M^e \bmod N_p$, where $0 \leq M < N$.

Decrypting ciphertext C into plaintext M as $M = C^d \bmod N$.

Example #3

Generating prime number p and choosing the members of set S as {p,p}, and setting $N = p^2$.

It is preferred that p is set to the minimum bit length given existing security constraints and expected message size.

Calculating e as a small prime number, such as 65537.

Creating the set S_p as a proper superset of set S containing members {p,p,q}, and calculating $N_p = p^2q$. It is preferred that q is large enough so that the bit length of the N_p reaches its recommended size.

Calculating the private exponent d such that $e*d = 1 \bmod (p - 1)$.

Encrypting plaintext M into ciphertext C as $C = M^e \bmod N_p$, where $0 \leq M < N$.

Decrypting ciphertext C into plaintext M by:

Precomputing the value $e\_inv\_p = e^{-1} \bmod p$;

Calculating $C_s = C \bmod p^2$;

Calculating $M_1 = C_s^{d-1} \bmod p$;

Calculating $K_0 = (M_1 * C_s) \bmod p$;

Calculating $A = (C - K_0^e) \bmod p^2$;

Calculating $M_2 = (M_1 * A) \bmod p^2$;

Calculating $M_3 = (M_2 * e\_inv\_p) \bmod p^2$;

Decoding plaintext message $M = (M_3 + K_0) \bmod p^2$;

<u>Example #4</u>

Generating distinct prime numbers p and q, and choosing the members of set S as {p,q}, and setting $N = p*q$.

Calculating e as a small prime number, such as 65537.

Creating the set S_p as a proper superset of set S containing members {p,q,r}, and calculating $N_p = pqr$, with q chosen so that that N is a squarefree number (all prime factors are distinct).

Calculating the private exponent d such that $e*d = 1 \bmod (p-1)(q-1)$.

Encrypting plaintext M into ciphertext C as $C = M^e \bmod N_p$, where $0 \leq M < N_p$.

Decrypting ciphertext C into plaintext M by:

Calculating $M_p = M \bmod p$;

Calculating $M_q = M \bmod p$;

Calculating $p\_inv\_q = p^{-1} \bmod q$;

Calculating $V = (M_q - M_p) \bmod q$;

Calculating $V_1 = V * p\_inv\_q \bmod q$;

Calculating $M_1 = V * p \bmod N$;

Calculating plaintext $M = (M_1 + M_p) \bmod N$;

Example #5

Being provided a public key {e, N};

Generating a number X as a large prime number;

Encrypting a plaintext message M into a ciphertext message C as: $C = M^e \bmod (N*X)$;

Provided that $M < X$ and $M < N$, decryption can occur in either of two ways:

$M = C^d \bmod N$, using the private key that corresponds to the public key {e, N}

Or

$M = C^{d_x} \bmod X$, where d_x is calculated such that $e * d_x = 1 \bmod (X - 1)$

Certain modifications and improvements will occur to those skilled in the art upon a reading of the foregoing description. All modifications and improvements have been deleted herein for the sake of conciseness and readability but are properly within the scope of the following claims.